Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

# APPLICATIONS INTERNET

## PROTOCOLS FOR THE INTERNET OF THINGS – PART II

SERGE AYER - HEIA-FR – TÉLÉCOMMUNICATIONS
CLASSES ISC-2D // 2023-2024

# COAP OPTIONS

- May be included in both requests and responses
- An option is never mandatory
- An option is identified by an option number
- Falls into one of two classes:
  - Critical (odd option number)
    - Must cause a 4.02 (Bad Option) response – with a diagnostic payload - when options of a CON request are not recognized by the destination endpoint.
    - Must cause a NON message to be rejected if options are unrecognized by the destination endpoint
    - Must cause a response or an ACK message to a CON request to be rejected if options are unrecognized by the source endpoint of the original request.

# COAP OPTIONS

- Elective (even option number)
  - Must be silently ignored if not recognized
- Also classified on how a proxy has to deal with the option
  - Unsafe to forward or Safe to forward
  - For Safe to forward, the option number also specified whether it is intended or not to be part of the Cache-Key
- An option may be repeatable or not
  - An option that is repeatable MAY be included one or more times in a message.
  - An option that is not repeatable MUST NOT be included more than once in a message.

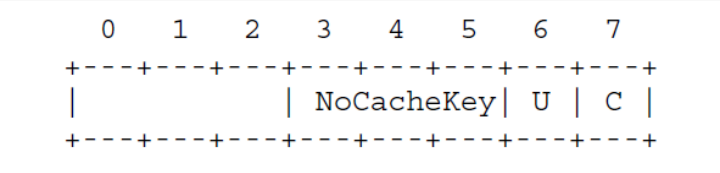
# COAP OPTIONS

| No. | C | U | N | R | Name | Format | Length | Default |
|---|---|---|---|---|---|---|---|---|
| 1 | x | | | x | If-Match | opaque | 0-8 | (none) |
| 3 | x | x | - | | Uri-Host | string | 1-255 | (see below) |
| 4 | | | | x | ETag | opaque | 1-8 | (none) |
| 5 | x | | | | If-None-Match | empty | 0 | (none) |
| 7 | x | x | - | | Uri-Port | uint | 0-2 | (see below) |
| 8 | | | | x | Location-Path | string | 0-255 | (none) |
| 11 | x | x | - | x | Uri-Path | string | 0-255 | (none) |
| 12 | | | | | Content-Format | uint | 0-2 | (none) |
| 14 | | x | - | | Max-Age | uint | 0-4 | 60 |
| 15 | x | x | - | x | Uri-Query | string | 0-255 | (none) |
| 17 | x | | | | Accept | uint | 0-2 | (none) |
| 20 | | | | x | Location-Query | string | 0-255 | (none) |
| 35 | x | x | - | | Proxy-Uri | string | 1-1034 | (none) |
| 39 | x | x | - | | Proxy-Scheme | string | 1-255 | (none) |
| 60 | | | x | | Size1 | uint | 0-4 | (none) |

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

# COAP OPTIONS

- Option number mask



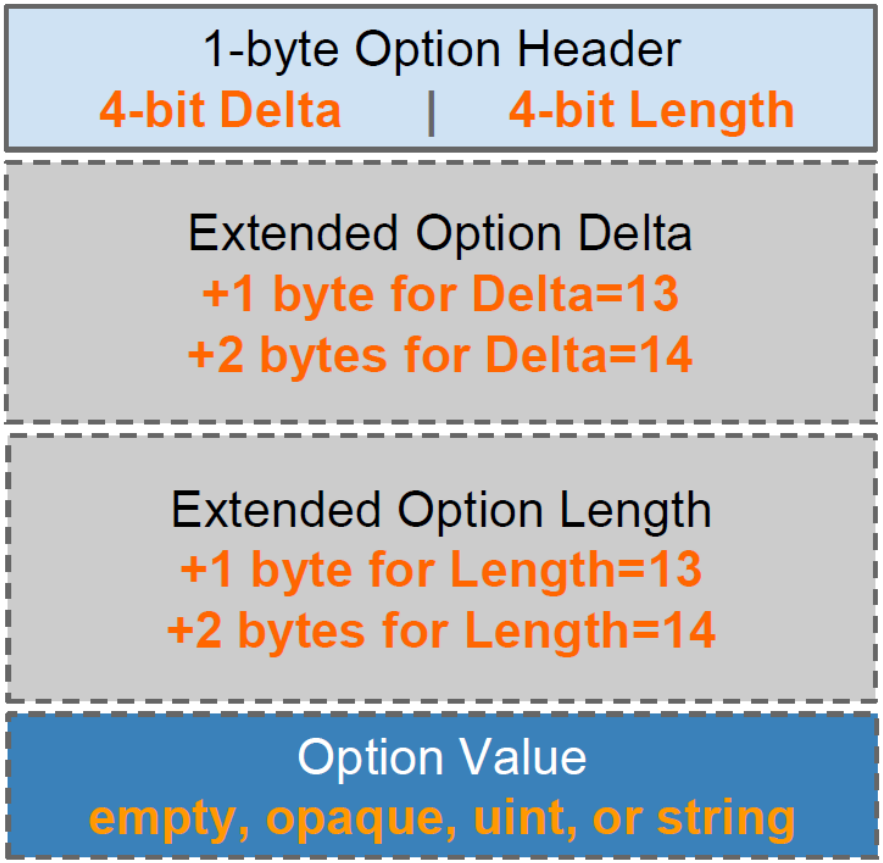

- Option value format
  - empty: a zero-length sequence of bytes
  - opaque: an opaque sequence of bytes
  - uint: a non-negative integer, represented in network byte order (big-endian), using the number of bytes given by the Option Length field.
  - Important note: it may be defined for a specific option that the uint value is represented using a range of permissible numbers of bytes, meaning that the sender should represent the integer with as few bytes as possible (without leading zeros – so 0 represented by an empty option value)
  - string: Unicode string encoded using UTF-8

# URI RELATED OPTIONS

- Unlike HTTP, the uri is specified in different options
- CoAP URI scheme
  - coap-URI = "coap:" "//" host [ ":" port] path-abempty [ "?"query ]
- Default values:
  - Uri-Host: IP literal representing the destination IP-address
  - Uri-Port: destination UDP port
- Repeated Uri-path:
  - Each specifies one segment of the absolute path to the resource
- Repeated Uri-Query:
  - Each specifies one argument parameterizing the request
- Proxy-Uri / Proxy-Scheme:
  - Used to make a request to a forward-proxy

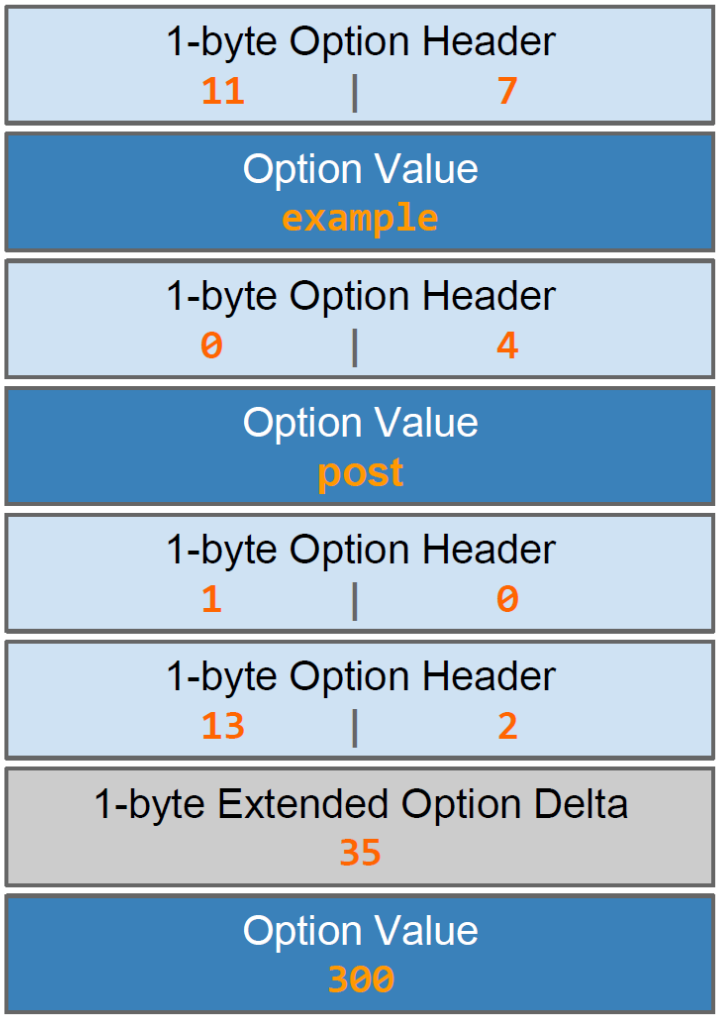
# COAP OPTION STRUCTURE

- Delta encoding
  - Option numbers are not encoded directly
  - They must appear in order and delta encoding is used
  - Delta == 13
    - Extended option is a 8-bit integer (delta value = 13 + extended option value)
  - Delta == 14
    - Extended option is a 16-bit integer (delta value = 269 + extended option value)
  - Delta == 15
    - Reserved for payload marker or future use



# COAP OPTION EXAMPLE

- Option delta = 0
- First option has number 11
  - Uri-path of length 7
  - Uri-path value = example
- Second option has number 11
  - (Repeated) Uri-path of length 4
  - Uri-path value = post
- Third option has number 11 + 1
  - Content-format of length 0 = value 0
  - Content-format value = text/plain
- Fourth option has an extended value
  - Delta value = 35 + 13 = 48
  - Option number = 60
  - Size1 with value 300, used in 4.13 responses for indicating the maximum size of the request entity



# COAP-TO-COAP PROXYING

- CoAP endpoint that performs requests on behalf of a client
  - Useful for using a cache for providing responses
- Proxies can be
  - Forward-proxy: explicitly chosen by the client
    - The request URI is specified in the Proxy-Uri option, or
    - The request URI is assembled from the Proxy-Scheme option and the Uri-* options.
    - The origin server's IP address and port are determined by the authority component of the request URI.
    - The Proxy-Uri or Proxy-Scheme is not forwarded to the origin server.
  - Reverse-proxy: inserted to stand in for origin servers
    - No use of the Proxy-Uri or Proxy-Scheme
    - Determines the destination of a request from information in the request and in its configuration

# COAP PAYLOAD AND REPRESENTATION

- REST means exchanging resource state in form of representations of a Web resource
- In CoAP, representations are carried as message payload in most responses as well as POST and PUT requests
- HTTP Content-Type and Content-Encoding options are merged into the single Content-Format option
  - Format is encoded as a number
  - No default value !

| Media type | Encoding | ID | Reference |
|---|---|---|---|
| text/plain; charset=utf-8 | - | 0 | [RFC2046] [RFC3676] [RFC5147] |
| application/link-format | - | 40 | [RFC6690] |
| application/xml | - | 41 | [RFC3023] |
| application/octet-stream | - | 42 | [RFC2045] [RFC2046] |
| application/exi | - | 47 | [REC-exi-20140211] |
| application/json | - | 50 | [RFC7159] |

# COAP PAYLOAD AND REPRESENTATION

- Optimized content format negotiation:
  - Through the Accept option used to indicate which Content-Format is acceptable to the client.
  - The Accept option is critical and not repeatable.
    - For easier cache implementation.
  - If no Accept option is given, the client does not express a preference – no default value ! – and prefers the representation chosen by the server returned in the Content-Format option.
  - If an Accept option is given, the server must return an error response with code 4.06 Not Acceptable or 4.02 Bad Option.

# COAP EXTENSIONS

- CoAP conception is modular
  - Endpoints only need to implement the features that they actually require
- Several extensions have been specified:
  - Resource Observation
    - Implements an efficient server push notification mechanism
  - Advanced congestion control (CoCoA)
    - Use round-trip delay measurements (RTT) measurements for adapting the retransmission time outs (RTOs), without re-inventing TCP

# COAP EXTENSIONS

- Blockwise transfers
  - Split the payload into multiple chunks
  - Useful for transferring large data while avoiding IP and 6LoWPAN fragmentation
  - Useful for incremental generation of large resource representations on constrained devices
  - Commonly used for resource discovery
- Alternative transports
  - Proposal for bindings for the Short Message Service (SMS)

# OBSERVING RESOURCES IN COAP

- A key feature for the IoT or WoT !
  - Enables efficient server push notifications.
- Designed as an optional feature on top of GET with an elective Observe option
  - Set by 0 by the client
  - If the server does not support it, fallback to normal GET and client polling.

# OBSERVING RESOURCES IN COAP

- If the server supports it, it will respond with this option, which turns the response into a notification.
  - The server will keep the client on its list of observers
  - The server will push new representations whenever the observed resource changes
  - The request/response pattern is transformed into a request/multiple-response pattern (using all the same token).
- Notifications can make use of cache control
  - Have a valid lifetime defined by the Max-Age option
  - Usually, the server sends a new representation before Max-Age expires.
  - Are cacheable
- Observations may be cancelled re-actively or pro-actively

# COAP BLOCK TRANSFER

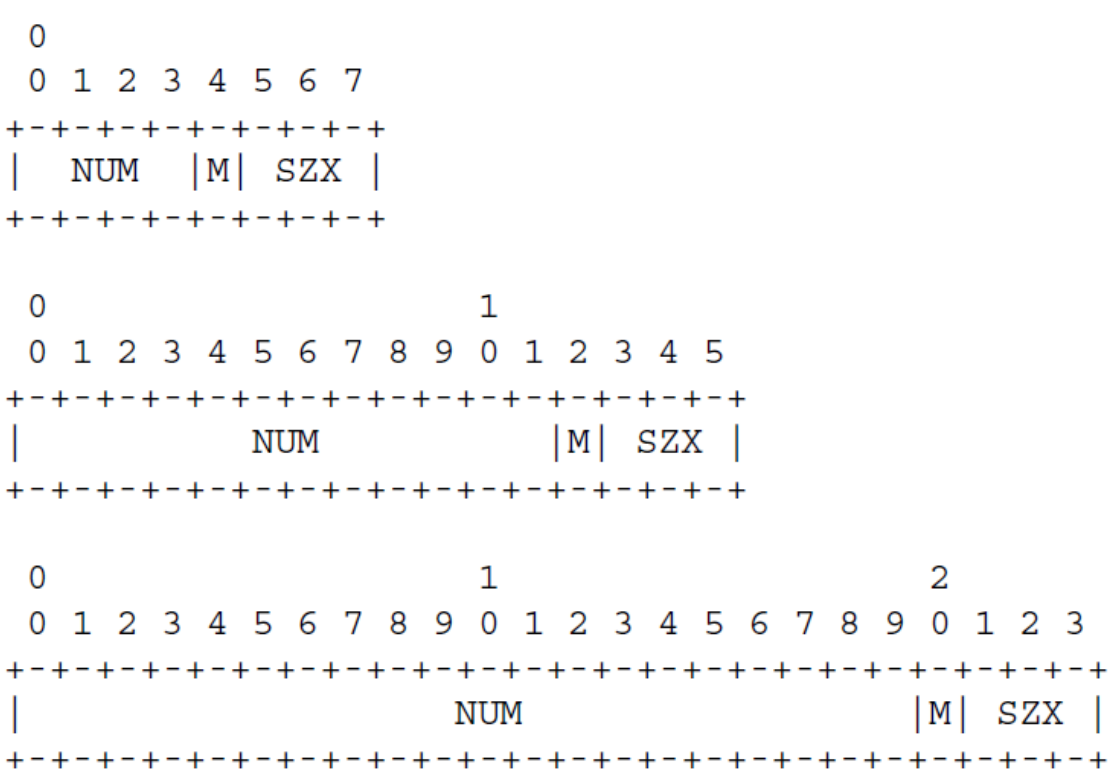
- Implemented with two additional, new options Block1 and Block2

| No. | C | U | N | R | Name | Format | Length | Default |
|---|---|---|---|---|---|---|---|---|
| 23 | C | U | - | - | Block2 | uint | 0-3 | (none) |
| 27 | C | U | - | - | Block1 | uint | 0-3 | (none) |

  - Can be present both in request and response messages
  - Block1 pertains to the request payload
    - Useful for PUT and POST requests and their responses
  - Block2 pertains to the response payload
    - Useful for GET, POST, and PUT requests and their responses

# COAP BLOCK TRANSFER

- Contains the size of the block (SZX), whether more blocks are following (M) and the relative number of the block (NUM)
- The actual Block size is 2^(SZX+4)



# COAP SERVICE DESCRIPTION AND DISCOVERY

- Using REST, all resources are provided through Web resources that are accessed through an initial URI, so
  - There is a machine-readable description of resources,
  - There is a mechanism to discover them on a server, and
  - There are ways to discover individual devices.
- Resource discovery
  - CoAP servers provide the CoRE Link Format separately as resource representation in its own Internet Media Type (application/link-format).
  - The initial URI on servers is /.well-known/core(RFC 5785)
  - The Link Format provided here can list all resources provided by the server as relative Web links (e.g., /sensors/temp),
  - It can also link to other service entry point URIs that provide their own Link Format.
  - In other words, the discovery mechanism of CoAP is hypermedia-driven, fulfilling the HATEOAS constraint of REST.

# COAP SERVICE DESCRIPTION AND DISCOVERY

- Device discovery is possible in two ways
  - Inquire the well-known discovery resource through the ‘all CoAP nodes’ multicast address,
    - This yields multiple responses with the CoRE Link Format.
    - This is limited to link-local and site-local networks.
    - This might cause congestion through numerous responses.
    - Thus, multicast discovery should only be used with filtering and response suppression by non-matching devices.

# COAP SERVICE DESCRIPTION AND DISCOVERY

- Device discovery is possible in two ways
  - Using a resource directory
    - Requires an additional entity, the CoRE resource directory (RD),
    - More scalable and reliable,
    - On start-up, devices register their resources (or selected service entry points) by POSTing their Link Format description to the RD, together with metadata such as a specific endpoint name or the lifetime of the registration.
    - Registration works as a heartbeat interval with which the device needs to report in: if there is no re-registration within this time, the device will be removed from the RD.

# INTEGRATION OF COAP WITH EXISTING INFRASTRUCTURES

- Need for a seamless integration into the existing Web infrastructure.
- In particular, a transparent mapping to HTTP is required.
  - When existing HTTP-based applications require IoT support, they can simply use a cross-proxy thatconverts between the two Web protocols.
  - The RESTful methods, response codes, and media types have a one-to-one mapping.

# INTEGRATION OF COAP WITH EXISTING INFRASTRUCTURES

- Thus, unlike application-level gateways that do protocol translation, these proxies are application-agnostic.
  - They can remain untouched when the application is upgraded or new device types are added.
- CoAP being in-line with the end-to-end arguments of the Internet is the key difference to previous gateways that have been connecting IoT technology to the Internet.

# INTEGRATION OF COAP WITH EXISTING INFRASTRUCTURES